SQL Programming 2

- Join
  - –NATURAL JOIN,
  - –JOIN ... ON …
- Subquery
  - – EXISTS, NOT EXISTS, IN, NOT IN
  - –ANY, ALL
- Readings: Sections 6.2.1— 6.2.3, 6.3 of textbook.

Multi-relation Queries

- Interesting queries often combine data from more than one relation.
- There are several ways to compose such queries in SQL.
  - Join: Listing all relations in the FROM clause.
  - Subquery: A subquery is nested inside the WHERE (or FROM) clause of a query.

The Village Cinema Database

The Village Cinema database:

Movie(mvID, Title, Rating, Rel_date, Length, Studio)

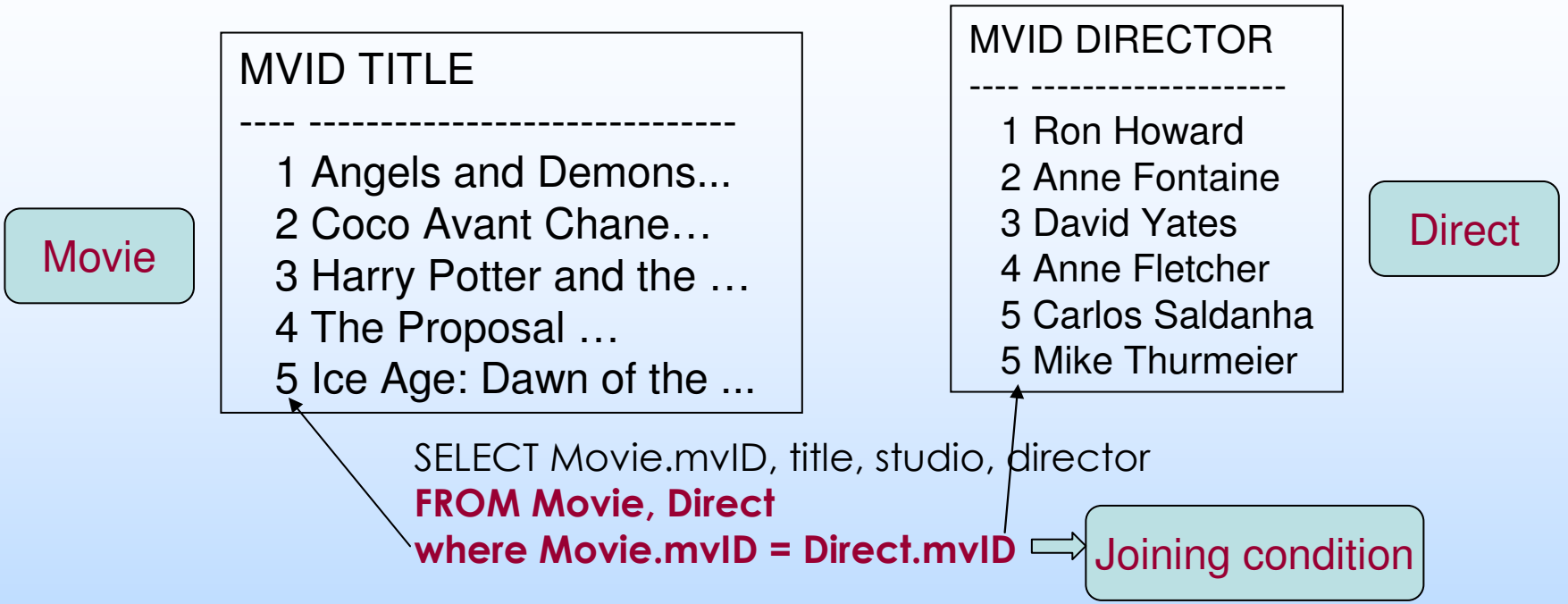
Classification(mvID*, Genre)

Cast(mvID*, Actor)

Direct(mvID*, Director)

All our SQL queries will be based on the Village Cinema database schema.

Joining Two Relations



MVID	TITLE	STUDIO	DIRECTOR
1	Angels and Demons	Sony Pictures	Ron Howard
2	Coco Avant Chanel	Roadshow	Anne Fontaine
3	Harry Potter and the Half-Blood Prince	Roadshow	David Yates
4	The Proposal	Disney	Anne Fletcher
5	Ice Age: Dawn of the Dinosaurs	20th Century Fox	Carlos Saldanha
5	Ice Age: Dawn of the Dinosaurs	20th Century Fox	Mike Thurmeier

The Joining Process

1. Combine every tuple in the first relation with every tuple in all other relations in the FROM clause.
2. Apply the joining condition from the WHERE clause.
3. Project onto the list of attributes and expressions in the SELECT clause.

Cartesian Product: Combining two relations

- For several relations in the FROM clause, without any WHERE clause, combinations of tuples in each relation are in the result – called the **Cartesian Product** of the relations.
- Example: There are 5*6=30 tuples (shown on the next slide) in the following cartesian product query:

```
SELECT Movie.mvID, Movie.title, Direct.mvID, Direct.director
FROM Movie, Direct
```

Only 6 tuples out of these 30 tuples satisfying

```
WHERE Movie.mvID=Direct.mvID.
```

Movie

MVID	TITLE
1	Angels and Demons...
2	Coco Avant Chane…
3	Harry Potter and the …
4	The Proposal …
5	Ice Age: Dawn of the ...

MVID	DIRECTOR
1	Ron Howard
2	Anne Fontaine
3	David Yates
4	Anne Fletcher
5	Carlos Saldanha
5	Mike Thurmeier

Direct

Cartesian Product …

MVID	TITLE	MVID	DIRECTOR
1	Angels and Demons	1	Ron Howard
1	Angels and Demons	2	Anne Fontaine
1	Angels and Demons	3	David Yates
1	Angels and Demons	4	Anne Fletcher
1	Angels and Demons	5	Carlos Saldanha
1	Angels and Demons	5	Mike Thurmeier
2	Coco Avant Chanel	1	Ron Howard
2	Coco Avant Chanel	2	Anne Fontaine
2	Coco Avant Chanel	3	David Yates
2	Coco Avant Chanel	4	Anne Fletcher
2	Coco Avant Chanel	5	Carlos Saldanha
2	Coco Avant Chanel	5	Mike Thurmeier
3	Harry Potter and the Half-Blood Prince	1	Ron Howard
3	Harry Potter and the Half-Blood Prince	2	Anne Fontaine
3	Harry Potter and the Half-Blood Prince	3	David Yates
3	Harry Potter and the Half-Blood Prince	4	Anne Fletcher
3	Harry Potter and the Half-Blood Prince	5	Carlos Saldanha
3	Harry Potter and the Half-Blood Prince	5	Mike Thurmeier
4	The Proposal	1	Ron Howard
4	The Proposal	2	Anne Fontaine
4	The Proposal	3	David Yates
4	The Proposal	4	Anne Fletcher
4	The Proposal	5	Carlos Saldanha
4	The Proposal	5	Mike Thurmeier
5	Ice Age: Dawn of the Dinosaurs	1	Ron Howard
5	Ice Age: Dawn of the Dinosaurs	2	Anne Fontaine
5	Ice Age: Dawn of the Dinosaurs	3	David Yates
5	Ice Age: Dawn of the Dinosaurs	4	Anne Fletcher
5	Ice Age: Dawn of the Dinosaurs	5	Carlos Saldanha
5	Ice Age: Dawn of the Dinosaurs	5	Mike Thurmeier

Each row of the Movie table is combined with every row of the Direct table!!

There are a lot of tuples. But the information on the title of a movie and the name of its matching director(s) is lost.

Explicit Tuple-Variables

- Sometimes a query needs to use two copies of the same relation.
- Distinguish copies by following the relation name by the name of a tuple-variable, in the FROM clause.

A Previous Example

- Which movies have both “Marie Gillain” and “Audrey Tautou”?

```
SELECT mvID
FROM cast
WHERE actor='Marie Gillain'
      AND actor ='Audrey Tautou';
```

Solution: Self-join

Cast C1

MVID	ACTOR
1	Tom Hanks
2	Alessandro Nivola
2	Audrey Tautou
2	Benolt Poelvoorde
2	Marie Gillain
3	Daniel Radcliffe
3	Emma Watson
3	Rupert Grint
4	Betty White
4	Malin Akerman
4	Mary Steenburgen
4	Ryan Reynolds
4	Sandra Bullock
5	Chris Wedge
5	Denis Leary
5	John Leguizamo
5	Queen Latifah
5	Ray Ramono

Cast C2

MVID	ACTOR
1	Tom Hanks
2	Alessandro Nivola
2	Audrey Tautou
2	Benolt Poelvoorde
2	Marie Gillain
3	Daniel Radcliffe
3	Emma Watson
3	Rupert Grint
4	Betty White
4	Malin Akerman
4	Mary Steenburgen
4	Ryan Reynolds
4	Sandra Bullock
5	Chris Wedge
5	Denis Leary
5	John Leguizamo
5 Queen Latifah
5	Ray Ramono

```
SELECT C1.mvID
FROM Cast C1, Cast C2
WHERE C1.mvID=C2.mvID
   AND C1.actor='Marie Gillain'
   AND C2.actor='Audrey Tautou';

MVID
--------
   2
```

Natural Join

- Natural join: Tuples from two relations are joined conditioned on that they have matching values for the common attributes.
- Common attributes appear only once in the result of Natural Join.

Natural Join ...

List all movies, including their mvID, title rating, studio and director information.

```
SELECT  mvID, title, rating, studio, director
FROM Movie natural join Direct
```

MVID	TITLE	RA	STUDIO	DIRECTOR
1	Angels and Demons	M	Sony Pictures	Ron Howard
2	Coco Avant Chanel	PG	Roadshow	Anne Fontaine
3	Harry Potter and the Half-Blood Prince	M	Roadshow	David Yates
4	The Proposal	PG	Disney	Anne Fletcher
5	Ice Age: Dawn of the Dinosaurs	PG	20th Century Fox	Carlos Saldanha
5	Ice Age: Dawn of the Dinosaurs	PG	20th Century Fox	Mike Thurmeier

Natural Join ...

Are the two queries below equivalent?

- Almost “yes” but with subtle difference --- they have the same number of tuples in the output but subtle difference in the number of attributes. What is it?

```
SELECT *
FROM Movie NATURAL JOIN Direct
```

mvID is included once in the output.

```
SELECT *
FROM Movie, Direct
WHERE Movie.mvID = Direct.mvID
```

Both Movie.mvID and Direct.mvID are included in the output.

Theta Join

- Theta join: Join two relations on any condition:

  <*relation 1*> JOIN <*relation 2*> ON *Condition.*

Theta Join: Example

- Find the movies that have at least two directors.

```
select distinct D1.mvID
from Direct D1 JOIN Direct D2 on
      D1.mvID = D2.mvID and D1.Director <> D2.Director
```

Direct

MVID	DIRECTOR
1	Ron Howard
2	Anne Fontaine
3	David Yates
4	Anne Fletcher
5	Carlos Saldanha
5	Mike Thurmeier

Output

MVID
5

Avoid using NATURAL JOIN in Oracle

- The joining condition for the Oracle NATURAL JOIN operator is implicitly specified.
- Generally the NATURAL JOIN operator Joins on all matching columns between two tables
    - Columns with the same name and type
    - The semantics of the columns are not considered
  - NATURAL JOIN multiple tables on multiple attributes can be problematic
  - Does not work on renamed relations (for relation self join)
- Try to avoid using NATURAL JOIN, use JOIN ... ON ... Instead. Although it means a bit more typing, you specify explicitly the joining condition.

Avoid using NATURAL JOIN ... In Oracle

Three relations:
A(A, Arest)
B(B, brest)
AB(A, B)

A	AREST
a1	a1 text
a2	a2 text
a3	a3 text
a4	a4 text

B	BREST
b1	b1 text
b2	b2 text
b3	b3 text
b4	b4 text

A	B
a1	b1
a2	b2

```
select *
from A natural join B;

A  AREST           B  BREST
-- ---------------- -- ----------------
a1 a1 text         b1 b1 text
a1 a1 text         b2 b2 text
a1 a1 text         b3 b3 text
a1 a1 text         b4 b4 text
a2 a2 text         b1 b1 text
a2 a2 text         b2 b2 text
a2 a2 text         b3 b3 text
a2 a2 text         b4 b4 text
a3 a3 text         b1 b1 text
a3 a3 text         b2 b2 text
a3 a3 text         b3 b3 text
a3 a3 text         b4 b4 text
a4 a4 text         b1 b1 text
a4 a4 text         b2 b2 text
a4 a4 text         b3 b3 text
a4 a4 text         b4 b4 text
16 rows selected.
```

A and B have no common attributes!

```
select *
from A natural join AB natural join B;

B  A  AREST           BREST
-- -- ---------------- ----------------
b1 a1 a1 text         b1 text
b2 a2 a2 text         b2 text

select count(*)
from A natural join AB natural join B;

  COUNT(*)
----------
         8
```

Join: A Complex Example

- Find the movies that have at least one genre that is the same as that of “Harry Potter and the Half-Blood Prince”.

```
select M2.mvID, M2.title, C2.genre
from movie M1, classification C1, movie M2, classification C2
where M1.mvID = C1.mvID
      and M1.title='Harry Potter and the Half-Blood Prince'
      and M2.mvID=C2.mvID
      and M1.mvID != M2.mvID
      and C1.genre=C2.genre
```

M1-C1 is about Harry Potter.

M2-C2 are about the movies to be found and output.

M2 should not be Harry Potter itself.

Join: A Complex Example

- Natural JOIN --- easier to understand.

```
select MC2.mvID, MC2.title, MC2.genre
from (select * from movie natural join classification) MC1,
       (select * from movie natural join classification) MC2
where
            MC1.mvID != MC2.mvID
     and MC1.genre=MC2.genre
     and MC1.title='Harry Potter and the Half-Blood Prince'
```

MC1

MVID	TITLE	GENRE
1	Angels and Demons	Drama
2	Coco Avant Chanel	Drama
3	Harry Potter …	Action
3	Harry Potter …	Adventure
3	Harry Potter …	Drama
4	The Proposal	Comedy
5	Ice Age …	Animated
5	Ice Age …	Comedy

MC2

MVID	TITLE	GENRE
1	Angels and Demons	Drama
2	Coco Avant Chanel	Drama
3	Harry Potter …	Action
3	Harry Potter …	Adventure
3	Harry Potter …	Drama
4	The Proposal	Comedy
5	Ice Age …	Animated
5	Ice Age …	Comedy

Output

MVID	TITLE	GENRE
1	Angels and Demons	Drama
2	Coco Avant Chanel	Drama

Exercise

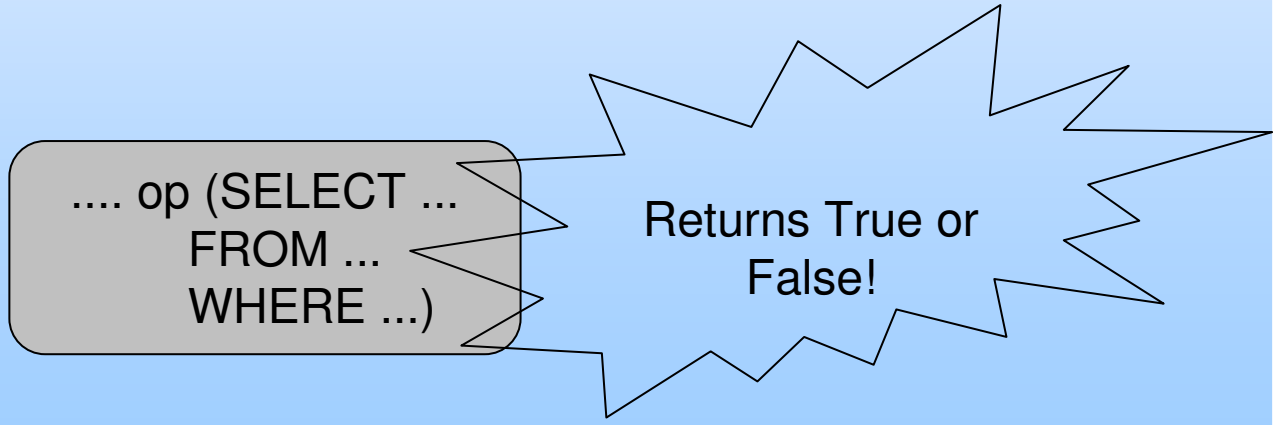
- What is the following query doing?

```
select distinct m1.mvID, m1.title
from movie m1 join movie m2
        on m1.length > m2.length
```

Subqueries

- A parenthesized SELECT-FROM-WHERE statement (*subquery*) can be used as a value in the WHERE (or FROM) clause of another query.

SELECT ...
FROM ...
WHERE ....



Subqueries that return a relation

- Generally a subquery returns a relation --- a set of tuples.
- To use the result of the subquery in the WHERE clause, we need operators (NOT) IN and (NOT) EXISTS. The result of IN and EXISTS expressions is TRUE or FALSE.

The IN Operator

- <tuple> IN (<subquery>) is true if and only if the tuple is a member of the relation produced by the subquery.
  - Opposite: <tuple> NOT IN (<subquery>).

Example: IN

- Find the movies that are dramas.

```
SELECT mvID, title
FROM Movie
WHERE mvID IN (SELECT mvID
               FROM Classification
               WHERE Genre='Drama')
```

Loops over tuples of Movie!

Loop over the tuples of Classification!

- Movie is the outer relation, and Classification is the inner relation.
- Only attributes of outer relation can be output.

Example: IN

Movie

MVID	TITLE
1	Angels and Demons
2	Coco Avant Chanel
3	Harry Potter and the Half-Blood Prince
4	The Proposal
5	Ice Age: Dawn of the Dinosaurs

```
SELECT mvID
FROM Classification
WHERE genre='Drama';
```

MVID
1
2
3

IN?

```
SELECT mvID, title
FROM Movie
WHERE mvID IN (SELECT mvID
                FROM Classification
                WHERE Genre='Drama')
```

MVID	TITLE
1	Angels and Demons
2	Coco Avant Chanel
3	Harry Potter and the Half-Blood Prince

Example: NOT IN

Movie

MVID	TITLE
1	Angels and Demons
2	Coco Avant Chanel
3	Harry Potter and the Half-Blood Prince
4	The Proposal
5	Ice Age: Dawn of the Dinosaurs

```
SELECT mvID
FROM Classification
WHERE genre='Drama';
```

MVID
1
2
3

NOT IN?

```
SELECT mvID, title
FROM Movie
WHERE mvID NOT IN (SELECT mvID
                   FROM Classification
                   WHERE Genre='Drama')
```

MVID	TITLE
4	The Proposal
5	Ice Age: Dawn of the Dinosaurs

The Exists Operator

- EXISTS(<subquery>) is true if and only if the subquery result is not empty.
  - Opposite: NOT EXISTS (<subquery>)

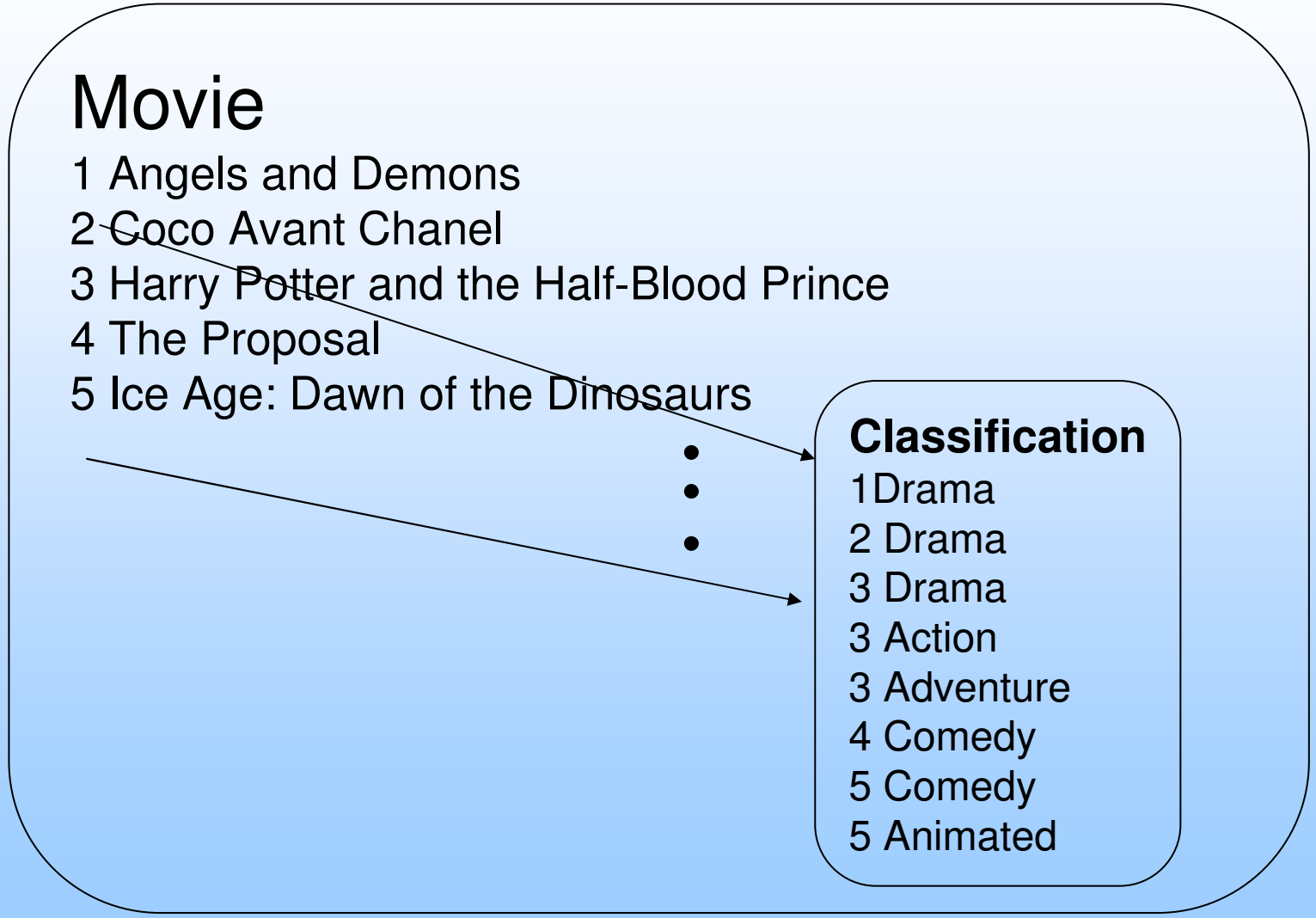
Example: EXISTS

```
SELECT mvID, title
FROM Movie
WHERE
    EXISTS (SELECT *
    FROM Classification
        WHERE mvID = Movie.mvID AND
            Genre = 'Drama');
```

Variable scope rule:
The inner-most relation by default.

Scope: The outer relation Movie.

Example: EXISTS



Example: NOT EXISTS

```
SELECT mvID, title
FROM Movie
WHERE
```

```
NOT EXISTS (SELECT *
FROM Classification
        WHERE mvID = Movie.mvID AND
                Genre = 'Drama');
```

Subqueries that returns a value

- When a subquery returns a value, the expression

  <value> = (<subquery>)

  can be used in the WHERE clause of a query.
  - Other relational operators can also be used, including <>, !=, >, >=, <, <=.
- Errors will occur if the relational operators are used for subqueries that do not return a value.

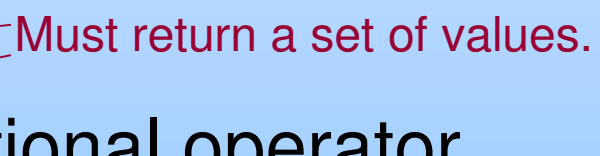
Subqueries that return a value: Example

The following query is correct only if it is known that Tom Hanks only performs in one movie.

```
select mvID, title
from Movie
where mvID = (select mvID
              from Cast
              where Actor='Tom Hanks');
```

The ANY and ALL operators

- In an comparison expression, the ANY/ ALL operator makes a set of values (returned by a subquery) comparable to a value. The expression is used in the WHERE clause of a query.
  - <value> = ANY (subquery)
  - <value > = ALL (subquery)
  - ANY (<subquery>) = <value>
  - ALL (<subquery>) = <value>
- In addition to =, any other relational operator can be used.



The Operator ANY

- *X*= ANY (*<a set>*) results in true iff there exists at least one value from the set that equals *X*.
- *X* = ANY(<subquery>) results in true iff *x* equals at least one value in the subquery result.
  - The subquery must return a set of values.

The operator ANY: Example

- What is the following query doing?

```
select mvID, title
from movie
where length > ANY(select length
                   from movie);
```

The Operator ALL

- *X* <> ALL(<subquery>) is true iff X is not equal to any value in the result retuned by the subquery.
  - That is, *x* is not in the subquery result.
- *X* > ALL(<subquery>) is true iff *X is* greater than all values returned by the subquery.

ALL: Example

- What is the following query doing?

```
select mvID, title
from movie
where length >= ALL(select length
                    from movie);
```

- What is result of the following query?

```
select mvID, title
from movie
where length > ALL(select length
                   from movie);
```

Exercise

- Find the movies whose length is larger than the average length of movies. Return the mvID, title and length of these movies.